Praktikum 02

8086 - Einführung Befehlssatz

Aufbau Assemblerbefehl

...wie ist ein Assemblerbefehl aufgebaut?

- inhaltliche Zweiteilung
Codeteil	Inhalt
Op-Code	auszuführende Operation
Operanden- und Adressfeld (OA-Feld)	zu verarbeitende Daten

Erläuterung Begriffe

...erläutern Sie die nachfolgenden Begriffe.
Begriff	Bedeutung
Maschinenbefehl	von Prozessor direkt ausführbarer Befehl
Op-Code	ist eine Zahl, die die Nummer eines Maschinenbefehls für einen bestimmten Prozessortyp angibt, alle Opcodes zusammen bilden den Befehlssatz des Prozessors/ der Prozessorfamilie
Daten	durch die CPU zu verarbeitende Informationen
Konstante	Behälter für einen Wert, der nach der Zuweisung nicht verändert werden kann
Variable	Behälter für Wert, der sich abhängig von den Bedingungen oder Informationen, die dem Programm übergeben werden, verändern kann/ welcher sich im Laufe des Programms verändern kann
Quelloperand (Source)	
Zieloperand (Destination)	

Funktionsweise Befehle

...wie funktionieren die nachfolgenden Befehle prinzipiell?

Befehlskürzel	Befehl	Befehlstyp	Funktion
MOV	MOV dest,source	Transportbefehl	Inhalt von *source* in *dest* kopieren
IN	IN dest,source	Eingabebefehl	von Datenport *source* wird Operand nach *dest* gelesen
OUT	OUT dest,source	Ausgabebefehl	Inhalt von *source* wird an Datenport *dest* ausgegeben

Befehlskürzel	Befehl	Befehlstyp	Funktion
ADD	ADD dest,source	Arithmetischer Befehl	Inhalte werden addiert, *dest* wird mit Ergebnis überschrieben

Befehlskürzel	Befehl	Befehlstyp	Funktion
SUB	SUB dest,source	Arithmetischer Befehl	*source* wird von *dest* subtrahiert (also *dest-source*), *dest* wird mit Ergebnis überschrieben
INC	INC dest	Arithmetischer Befehl	Inhalt *dest* wird um eins erhöht
DEC	DEC dest	Arithmetischer Befehl	Inhalt *dest* wird um eins erniedrigt
CMP	CMP dest,source	Vergleich	wie SUB, d.h. *dest-source*, aber *dest* wird nicht mit Ergebnis überschrieben, lediglich Flags werden beeinflusst

Befehlskürzel	Befehl	Befehlstyp	Funktion
NOT	NOT dest	Logischer Befehl	*dest* wird bitweise komplementiert/ invertiert
AND	AND dest,source	Logischer Befehl	Inhalte *dest* und *source* werden bitweise UND-verknüpft, *dest* wird mit Ergebnis überschrieben
OR	OR dest,source	Logischer Befehl	Inhalte *dest* und *source* werden bitweise ODER-verknüpft, *dest* wird mit Ergebnis überschrieben
XOR	XOR dest,source	Logischer Befehl	Inhalte *dest* und *source* werden bitweise XOR-verknüpft, *dest* wird mit Ergebnis überschrieben

Befehlskürzel	Befehl	Befehlstyp	Funktion
SHL	SHL source,count	Schiebebefehl	Inhalt von *source count*-mal nach links schieben (am weitesten links stehende setzt/setzt nicht CF)
SHR	SHR source,count	Schiebebefehl	Inhalt von *source* wird *count*-mal nach rechts geschoben (am weitesten rechts stehende setzt/setzt nicht CF)
ROL	ROL source,count	Rotationsbefehl	Inhalt von *source* wird *count*-mal nach links rotiert
ROR	ROR source,count	Rotationsbefehl	Inhalt von *source* wird *count*-mal nach rechts rotiert
RCL	RCL source,count	Rotationsbefehl	Inhalt von *source* wird *count*-mal links durch das CF rotiert
RCR	RCR source,count	Rotationsbefehl	Inhalt von *source* wird *count*-mal nach rechts durch das CF rotiert

Verwendung der Baugruppen des SBC86

Baugruppe	Verwendung
Abfrage der Schalterstellung	IN AL,0

Baugruppe	Verwendung
Direkte Ansteuerung der LED-Reihe	OUT 0,AL (Inhalt AL bestimmt über welche LEDs leuchten werden)
Direkte Ansteuerung 7-Segment-Anzeige	über Adressports 90h-9eh, Ausgabe von '1' lässt zugehörige Segment aufleuchten

Funktion und Wirkung einzelner Befehle des Programms

...ohne ähnliche Befehle wiederholend zu notieren.
Befehl	Funktion und Wirkung
MOV AL,01010101b	kopiere Konstante in Zielregister
MOV CX,AX	kopiere Inhalt aus Quellregister in Zielregister
MOV [0150h],AL	kopiere Inhalt aus Quellregister in Zielregister
ADD CH,CL	addiere Registerinhalte
DEC BX	Dekrementiere (erniedrigen um 1) Registerinhalt
NOT BH	Registerinhalt bitweise invertieren
OR DL,BL	Registerinhalte bitweise ODER-verknuepfen, Zielregister mit Ergebnis ueberschreiben
AND DH,11001100b	Konstante mit Registerinhalt bitweise UND-verknuepfen, Zielregister mit Ergebnis ueberschreiben
ROL AL,1	Registerinhalt 1-mal nach links rotieren
OUT 0,AL	Registerinhalt AL an Port ausgeben (Ausgabe/Ansteuerung LED)
LOOP schl	Registerinhalt CX dekrementieren, zu <shortlabel> springen solange CX != 0
SHR BL,CL	Registerinhalt wird "Inhalt des Registers CX mal" nach rechts geschoben
OUT 90h, AL	Ausgabe Registerinhalt von AL an Port (Ausgabe/Ansteuerung Segment des 7-Segment-Displays)

Warum gibt es Unterschiede in der Schreibweise zwischen den Assemblerbefehlen im Quelltext mit angezeigten Befehlen im Fenster Disassembled Code des SBC8086-Emulator?

- Unterschiede
  1. Angabe Zahlen
  2. Angabe <shortlabel> des LOOP-Befehls
- Begründung Unterschiede
  1. Compiler übersetzt Zahlen einheitlich in Hex-Format
  2. Compiler ersetzt <shortlabel> mit Adresse des Befehls in Speicher, den er als nächstes ausführen soll

Möglichkeit Wartefunktion zu implementieren

...damit Zeit vergeudet wird.

- notwendig, wenn wir blinken der LED wahrnehmen wollen

```
warte:
        XOR CX,CX
        schlf:
                loop schlf
```